

# METHOD FOR CONSTITUTING MULTI-OS

Publication number: JP2001236237

Publication date: 2001-08-31

Inventor: KIMURA SHINJI; ARAI TOSHIKI; SATO MASAHIDE;  
UMETSU TOSHIKAZU

Applicant: HITACHI LTD

Classification:

- international: G06F11/28; G06F9/06; G06F9/46; G06F11/28;  
G06F9/06; G06F9/46; (IPC1-7): G06F9/46; G06F9/06;  
G06F11/28

- European:

Application number: JP20000052108 20000223

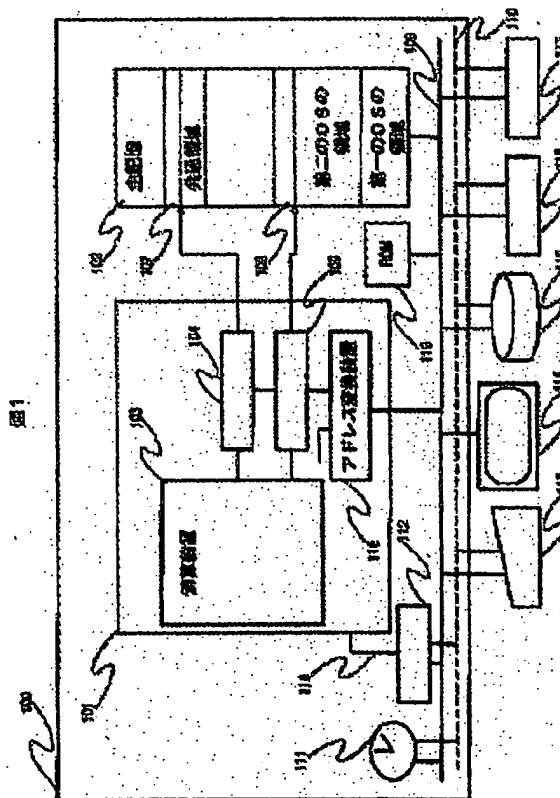
Priority number(s): JP20000052108 20000223

Report a data error here

## Abstract of JP2001236237

**PROBLEM TO BE SOLVED:** To provide a method for constituting a multi-OS for reducing the generation of overhead without necessitating any special hardware.

**SOLUTION:** A physical memory 102 is divided for each of plural operating systems. An interruption management program independent of the operating system accepts all outside interruption, and decides which the operating system interruption handler should be started according to an interruption factor. A timing in which the interruption handler should be started is decided according to the executed state of the operating system, and the interruption handler of each operating system is started based on that.



Data supplied from the esp@cenet database - Worldwide

(19) 日本国特許庁 (J P)

(12) 公開特許公報 (A)

(11) 特許出願公開番号

特開2001-236237

(P2001-236237A)

(43) 公開日 平成13年8月31日 (2001.8.31)

(51) Int.Cl. <sup>7</sup>	識別記号	F I	テーマコード (参考)
G 0 6 F 9/46	3 5 0	G 0 6 F 9/46	3 5 0 5 B 0 4 2
9/06	4 1 0	9/06	4 1 0 D 5 B 0 7 6
11/28		11/28	A 5 B 0 9 8

審査請求 未請求 請求項の数 5 O L (全 22 頁)

(21) 出願番号 特願2000-52108 (P2000-52108)

(22) 出願日 平成12年2月23日 (2000.2.23)

(71) 出願人 000005108

株式会社日立製作所

東京都千代田区神田駿河台四丁目6番地

(72) 発明者 木村 信二

神奈川県川崎市麻生区王禅寺1099番地 株

式会社日立製作所システム開発研究所内

(72) 発明者 新井 利明

神奈川県川崎市麻生区王禅寺1099番地 株

式会社日立製作所システム開発研究所内

(74) 代理人 100075096

弁理士 作田 康夫

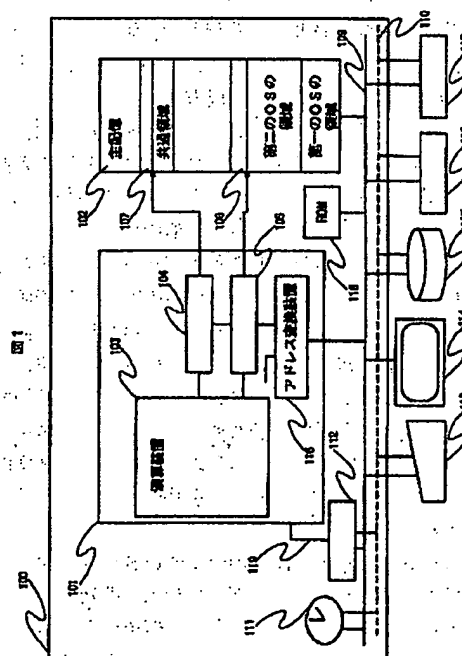
最終頁に続く

(54) 【発明の名称】 マルチOS構成方法

(57) 【要約】

【課題】 特殊なハードウェアを必要とせず、オーバーヘッドの発生を抑えたマルチOSの構成方法を提供する。

【解決手段】 複数のオペレーティングシステム毎に物理メモリ102を分割する。オペレーティングシステムから独立した割り込み管理プログラムが、すべての外部割り込みを受け付け、割り込み要因によりどのオペレーティングシステムの割り込みハンドラを起動すべきかを決定する。オペレーティングシステムの実行状態により割り込みハンドラを起動するタイミングを決定して、それに基づいて各オペレーティングシステムの割り込みハンドラを起動する。



## 【特許請求の範囲】

【請求項1】 計算機システムにおけるオペレーティングシステムの制御方法において、計算機のハードウェア資源を第一のオペレーティングシステムの管理対象から外して他のオペレーティングシステムに与える手順と、複数のオペレーティングシステムで共有する領域に配置した共通の処理手順とデータを有し、前記共通処理手順の割り込み処理部が各オペレーティングシステムの実行をスケジュールすることにより、各オペレーティングシステムが実行する特権命令のエミュレート処理なしで、複数のオペレーティングシステムを一台の計算機で動作させるようにしたマルチOSの構成方法において、複数のオペレーティングシステムで共有する領域に配置した共通の処理手順を第一のオペレーティングシステムのデバイスドライバとして組み込む手順と、該デバイスドライバが第一のオペレーティングシステムの割り込み処理部から前記共通処理手順の割り込み処理部へ制御を切り替える手順と、該デバイスドライバが第二のオペレーティングシステムをロードして第二のオペレーティングシステムとは別の仮想アドレス空間で第二のオペレーティングシステムを起動する手順とを有し、前記共通処理手順の割り込み処理部が各オペレーティングシステムの実行をスケジュールすることにより、第一のオペレーティングシステムの処理手順を変更することなしに、複数のオペレーティングシステムを一台の計算機で動作させることを特徴とするマルチOS構成方法。

【請求項2】 請求項1記載のマルチOS構成方法において、1つのオペレーティングシステムが使用する割り込み資源を他のオペレーティングシステムがアクセスできないようにするため、1つのオペレーティングシステムが割り込み資源を必要とする時点で、他のオペレーティングシステムに対しアクセスできないように予約することを特徴とするマルチOS構成方法。

【請求項3】 請求項1記載のマルチOS構成方法において、1つのオペレーティングシステムに割り込み資源を割り当て後、該オペレーティングシステムが割り込み資源を必要としなくなった時点で、他のオペレーティングシステムが該割り込み資源を利用できるようにするため、他のオペレーティングシステムの割り込み資源の予約を解除することを特徴とするマルチOS構成方法。

【請求項4】 請求項1記載のマルチOS構成方法において、複数のオペレーティングシステムで共有する領域に配置した共通処理手順及び共通データのアドレスを、すべてのオペレーティングシステムの仮想アドレス空間上で一致させる手順を有することを特徴とするマルチOS構成方法。

【請求項5】 請求項1記載のマルチOS構成方法において、被監視オペレーティングシステムの関数群のアドレス情報を取得する手順と、アドレス情報からオペレーティングシステム内で障害発生時にコールされるエラー関

数のアドレスを抽出する手順と、抽出したエラー関数のアドレスに割り込み発生命令を書き込む手順と、該割り込みが発生したとき、複数のオペレーティングシステムで共有する領域に配置した共通処理手順の割り込み処理部が監視オペレーティングシステムに通知する手順と、通知によって該監視オペレーティングシステムが被監視オペレーティングシステムの障害診断あるいは障害回復処理を行う手順を有することを特徴とするマルチOS構成方法。

## 【発明の詳細な説明】

## 【0001】

【発明の属する技術分野】 本発明は、一台の計算機上で複数のオペレーティングシステムを稼動させるためのマルチOS構成方法に関する。

## 【0002】

【従来の技術】 通常、計算機システムでは、計算機上で1つのオペレーティングシステムを動作させ、それにより計算機のプロセッサ、メモリ、および、二次記憶装置等の計算機資源を管理し、計算機が効率良く動作できるように資源スケジュールを実施している。オペレーティングシステムには様々な種類がある。バッチ処理に優れるものや、TSS (Time Sharing System) に優れるもの、GUI (Graphical UserInterface) に優れているものなど様々である。

【0003】 一方、これら複数あるオペレーティングシステムを一台の計算機で同時に実行したいというニーズがある。例えば、大型計算機においては、実際の業務に伴うオンライン処理を実行するオペレーティングシステムと、開発用のオペレーティングシステムを一台の計算機で動作させたいという要求がある。あるいは、GUIの整っているオペレーティングシステムと、実時間性に優れているオペレーティングシステムを同時に稼動させたいという要求もある。

【0004】 しかしながら、個々のオペレーティングシステムは、単独で計算機資源の管理を実施することを仮定しており、複数のオペレーティングシステムの共存は、何らかの機構なしに実現することは困難である。

【0005】 一台の計算機上で複数のオペレーティングシステムを動作させる機構として、大型計算機で実現されている仮想計算機方式がある。仮想計算機方式では、仮想計算機制御プログラムが全ハードウェア資源を占有して管理し、それを仮想化して仮想計算機を構成する。仮想計算機を構成する制御部は、物理メモリ、入出力機器装置、外部割り込み等を仮想化する。

【0006】 例えば、分割された物理メモリは、仮想計算機に対してはあたかも0番地から始まる物理メモリのように振る舞い、入出力装置を識別する装置番号も同様に仮想化されている。更に、仮想計算機方式では、磁気ディスクの記憶領域も分割して磁気ディスク装置の仮想化まで実現している。

10

20

30

40

50

【0007】各オペレーティングシステムは、制御プログラムにより構築された仮想計算機上で実行されるように制御プログラムによりスケジュールされる。

【0008】

【発明が解決しようとする課題】上述したような従来の大型計算機における仮想計算機方式では、計算機資源を完全に仮想化、および、分割しようとするため、仮想計算機を構成する制御部分が複雑になるという問題がある。

【0009】また、特別なハードウェア支援がない場合、仮想計算機上で動作するオペレーティングシステムが発行する制御レジスタの設定、入出力命令等の特権命令は、仮想計算機制御プログラムによりエミュレートしなければならない。このため、オーバーヘッドが大きくなるという問題もある。実際、仮想計算機を実装している大型計算機では、仮想計算機用に特別なプロセッサ機能やマイクロコード等のハードウェアを追加してオーバーヘッドの削減を図っている。

【0010】このように、仮想計算機方式は、完全に計算機資源を仮想化することを目的としているため複雑であり、更に、仮想計算機の高性能化のためには特殊なハードウェア機構が必要となる。

【0011】一方、一台の計算機上で複数のオペレーティングシステムのインタフェースを提供する技術として、マイクロカーネルと呼ばれる技術がある。マイクロカーネルでは、マイクロカーネルの上に、ユーザに見せるオペレーティングシステム機能を提供するオペレーティングシステムサーバが構築される。ユーザは、そのサーバを経由して計算機資源を利用する。オペレーティングシステム毎のサーバを用意すれば、ユーザに様々なオペレーティングシステム環境を提供できる。

【0012】しかし、マイクロカーネル方式では、オペレーティングシステムサーバをマイクロカーネルに合わせて新規に構築する必要がある。多くの場合、現在あるオペレーティングシステムをマイクロカーネル上で動作するように変更することになるが、スケジューリング、メモリ管理等のカーネルの中核部分も変更することになり、変更箇所が多く、また、変更箇所がオペレーティングシステムの中核部分に及ぶため変更作業が複雑で容易ではないといった問題がある。

【0013】また、オペレーティングシステムサーバはマイクロカーネルのサービスを利用することになるが、これは通常のオペレーティングシステムではないことであり、オーバーヘッドとなり性能低下をもたらす。

【0014】本発明の目的は、このような従来技術における問題点を解決し、特殊なハードウェアを用いることなく、1台の計算機上で複数のオペレーティングシステムの同時・並列的な実行を実現することにある。

【0015】また、本発明の他の目的は、1台の計算機上で複数のオペレーティングシステムを動作させるにあ

たり、特権命令のエミュレーションによるオーバーヘッドの発生を抑えることにある。

【0016】

【課題を解決するための手段】上述した目的を達成するために、本発明によるマルチOSの構成方法では、複数のオペレーティングシステム毎に物理メモリを分割する。オペレーティングシステムから独立した割り込み管理プログラムが、すべての外部割り込みを受け付け、割り込み要因によりどのオペレーティングシステムの割り込みハンドラを起動すべきかを決定する。オペレーティングシステムの実行状態により割り込みハンドラを起動するタイミングを決定して、それに基づいて各オペレーティングシステムの割り込みハンドラを起動することにより、複数のオペレーティングシステムを一台の計算機で動作させる。

【0017】本発明によれば、第一のオペレーティングシステムを補完する機能を容易に追加でき、高機能な計算機システムの構築が可能になる。更に、デバイスドライバとは異なり、第一のオペレーティングシステムとは全く独立して動作する機能を組み込むことができるため、第一のオペレーティングシステムに依存しない高信頼化機能を追加することも可能になる。

【0018】また、マイクロカーネル方式で複数のマルチオペレーティングシステム環境を構築する方法では、それぞれのオペレーティングシステムのインタフェースを提供するオペレーティングシステムサーバの構築が難しいという問題がある。本発明によれば、オペレーティングシステムを変更することなく、割り込み管理部分の追加だけで、簡単にマルチオペレーティングシステム環境を構成できる。

【0019】

【発明の実施の形態】図1は本発明の実施の形態における計算機100の構成を示すブロック図である。

【0020】計算機100は、プロセッサ101、主記憶装置102、バス109、割り込み信号線110、クロック割り込み生成器111、割り込み制御装置112、ブート手順を格納している記憶装置118、および、割り込みバス119を有して構成されている。

【0021】割り込み信号線110は、外部の入出力機器と割り込み制御装置112を接続している。外部機器が割り込みを発生すると、割り込み信号線110を經由して割り込み制御装置112が信号を受け取る。割り込み制御装置112は、この信号を数値化して、割り込みバス119を介してプロセッサ101に渡す。

【0022】クロック割り込み生成器111は、周期的な割り込みを生成する。

【0023】割り込み制御装置112は、外部機器からの割り込みを受け付け、要求元にしたがって数値化された割り込み信号を生成し、プロセッサ101に送る。また、プロセッサ101からの指示により、特定の機器か

らの割り込み信号をプロセッサ101に通知しないようにすることもできる。

【0024】プロセッサ101は、演算装置103、割り込みテーブルレジスタ104、ページテーブルレジスタ105、および、アドレス変換装置106を有して構成されている。

【0025】割り込みテーブルレジスタ104は、割り込みテーブル107の仮想アドレスを指し示している。割り込みテーブルの詳細については後述するが、割り込み番号毎の割り込みハンドラの開始アドレスを記録している。図1で、割り込みレジスタ104と割り込みテーブル107の接続を破線で記載しているのは、割り込みテーブルレジスタ104が割り込みテーブルの指し示すためである。

【0026】割り込みが発生すると、プロセッサ101は、割り込み制御装置112から数値化された割り込み番号を受ける。プロセッサ101は、この番号をインデックスとして割り込みテーブル107より割り込みハンドラアドレスを取得し、割り込みハンドラに制御を渡す。

【0027】ページテーブルレジスタ105は、ページテーブル108を指し示している。ページテーブルレジスタ105は、ページテーブル108の物理アドレスを格納している。

【0028】アドレス変換装置106は、演算装置が要求する命令アドレス、あるいは、オペランドが格納されているアドレスを受け取り、ページテーブルレジスタ105の指しているページテーブル108の内容に基づき仮想-実アドレス変換を実施する。

【0029】計算機100には、外部入出力装置としてキーボード113、ディスプレイ114、磁気ディスク115、その他の外部機器116、および、117が接続されている。ディスプレイ114を除く機器は割り込み信号線110により割り込み制御装置112に接続されている。

【0030】主記憶装置102の内容について簡単に説明する。ここでは、計算機101では2つのオペレーティングシステムが動作しているとする。それぞれを第一のオペレーティングシステム、第二のオペレーティングシステムと呼ぶことにする。また、計算機を起動すると第一のオペレーティングシステムが起動するように設定されており、外部機器116、および、117は第二のオペレーティングシステムにより管理される機器とする。以下では、第一のオペレーティングシステムを第一のOS、第二のオペレーティングシステムを第二のOSと記す。

【0031】第一のOSは、起動時の初期化処理で、その他のオペレーティングシステム用に、この場合は第二のOS用に物理メモリ領域を予約する。つまり、第一のOSが、第二のOS用に予約した物理メモリ領域を利用

できないように物理メモリ領域を確保する。図1は、この奪った領域に第二のOSがロードされている様子を示している。

【0032】また、主記憶装置102は、すべてのオペレーティングシステムから参照可能な共通領域を持つ。共通領域には、割り込みテーブル107、割り込み管理プログラム、割り込みハンドラ、各オペレーティングシステムから呼び出し可能なインタフェースモジュール等が格納される。

【0033】本実施形態の動作概要について説明する。本実施形態では、第二のOSは第一のOSよりも優先して動作する。優先して動作するとは、第一のOSは第二のOSがアイドル状態であるときのみ動作可能であることを示す。第二のOSの処理が終了しない限り、第一のOSは動作できない。

【0034】また、第二のOSが管理するデバイスが割り込みが発生すると、第一のOSの処理は中断され、制御は第二のOSに移る。第二のOS実行中に第一のOSへの割り込みが発生しても、その割り込み処理は第一のOSが実行されるまで延期される。

【0035】さらに、第一のOSと第二のOSは明確に区分されており、割り込みハンドラなどを配置する共通領域以外は、互いにアクセスできないように構成される。これにより、2つのオペレーティングシステムが誤って互いの領域にアクセスして障害が起きることを防いでいる。

【0036】図2は、本実施形態における2つのオペレーティングシステムの間接的な関係を示した模式図である。

【0037】それぞれのオペレーティングシステムは、それぞれ独立したアドレス空間を保持する。201は第一のOSの仮想空間で、202は第二のOSの仮想空間を示している。ここで、第二のOSの空間202に対応する実記憶は、図1の主記憶装置102の第二のOSの領域になる。

【0038】仮想空間の一部分には、共通領域203がマップされる。共通領域203に対応する実記憶は、図1において、主記憶102の共通領域として示される領域である。

【0039】図2は、また、各オペレーティングシステムが管理するハードウェアを示している。第一のOSは、キーボード113、ディスプレイ114、および、磁気ディスク115を、第二のOSは入出力装置116、および、117を管理する。クロック111と割り込み制御装置112は、もともとは第一のOSが管理しているハードウェアであるが、共通領域203中のプログラムが管理する。

【0040】図3は、本実施形態でのページテーブルの構成を示している。

【0041】ページテーブル300は、プロセッサ10

1の仮想アドレス空間の仮想ページ毎に、それぞれの仮想ページを記述するエントリを持っている。それぞれのエントリは、有効ビット301と、物理ページ番号302を含んで構成される。

【0042】有効ビット301は、その仮想ページに対応する物理ページが割り当てられているか、つまり、仮想-実アドレス変換が可能かを示している。例えば、ページテーブル300の仮想ページ3は、有効ビットがセットされていないので、仮想ページ3に対応する物理ページが存在しないことを示している。有効ビット301がセットされていない仮想ページへのアクセスが発生すると、プロセッサはページフォルトを発生する。

【0043】物理メモリ番号302は、仮想ページに対応する物理ページ番号を記録している。

【0044】アドレス変換装置106は、ページテーブルレジスタ105の指し示しているページテーブルの内容を参照して、演算装置103の生成する仮想アドレスを実アドレスに変換する。プロセッサ101は、変換により得られた実アドレスにより主記憶装置102を参照する。

【0045】ページテーブルを切り替えることにより独立した空間を構築することができ、図2に示した第一のOSの空間、および、第二のOSの空間の構築が可能である。また、共通領域203については、両方のオペレーティングシステムのページテーブルの共通領域に対応する部分に、同じ物理ページをマップするように設定しておけば、共通領域を実現できる。

【0046】図4は、割り込みテーブルの構成を示している。

【0047】割り込みテーブル400は、プロセッサ101が割り込み制御装置112から受ける割り込み番号毎の、割り込みハンドラの仮想アドレス401を記録している。プロセッサ101は、割り込み要求を割り込み制御装置112から受け取ると、割り込み番号に対応する割り込みハンドラのアドレスを、割り込みテーブルレジスタ104の指し示している割り込みテーブル400から取得し、そのアドレスに制御を移すことで割り込み処理を開始する。

【0048】図5は、割り込み制御装置112の構成を示している。

【0049】割り込み制御装置112は、割り込みマスクレジスタ501、および、選択装置502を持っている。

【0050】割り込みを発生する入出力装置は、割り込み信号線110により割り込み制御装置112と接続される。入出力機器の発生する割り込みは、割り込み信号線110のどの信号線に接続するかにより優先順位が付けられる。ここでは、割り込み0番に対応する割り込み信号がもっとも優先度が高い割り込みであるとする。

【0051】割り込み信号線110は、選択装置502

に接続されている。選択装置502は、割り込み信号を受けると、プロセッサがその割り込みを受けたことを通知するまで、未処理の割り込みがあることを記録している。

【0052】割り込みマスクレジスタ501は、入出力機器の発生した割り込みをプロセッサに通知してよいかを記録している。割り込みマスクレジスタ501は、プロセッサ101からの入出力命令により設定可能である。

【0053】選択装置502は、割り込み信号線110からの割り込み要求を受けた時と、割り込みマスクレジスタ501の内容が書き換えられた時に、選択装置502が記録している未処理割り込みと、割り込みマスクレジスタ502の内容を比較して、プロセッサに割り込みを通知するかどうかを決める。具体的には、選択装置502が記録している未処理割り込みのうち、割り込みマスクレジスタ501に割り込み可能と設定されていて、優先度の最も高い割り込みから順にプロセッサに通知する。選択した割り込みについて、選択装置502は、通知する割り込み信号に対応する数値信号を、割り込みバス119経由でプロセッサ101に送る。

【0054】プロセッサ101は、割り込みを受けたときに、入出力命令により選択装置502に記録されている未処理割り込みを解消できる。

【0055】次に、本実施形態における、計算機のブート手順について説明する。

【0056】ブート手順の始めの部分は、読み取り専用の記憶装置118に格納されている。記憶装置118は、プロセッサの物理アドレス空間のある決められたアドレスにマップされるように、バス109を介してプロセッサ101に接続されている。この手順は、ハードウェア構成の検出、オペレーティングシステムをロードするプログラムの主記憶へのローディングを実施する。

【0057】プロセッサ101がリセットされると、プロセッサ101は予め定められた物理アドレスに制御を移す。記憶装置118は、この時に実行されるプログラムを格納しており、プロセッサ101がリセットされた時にこのプログラムに制御を渡せるように物理アドレス空間にマップされている。

【0058】記憶装置118に格納されているプログラムは、磁気ディスク装置112に格納されている第一のOSのカーネルローダを主記憶装置102にロードして実行する。カーネルローダは、磁気ディスク装置112の予め定められた位置にあり、記憶装置118に格納されていたプログラムは、容易にこれを見つけることができる。

【0059】図6は、本実施形態における、オペレーティングシステムのカーネルローダの処理手順を示すフローチャートである。

【0060】カーネルローダは、オペレーティングシス

テムのファイルシステム構造を理解して、ファイル名によりファイルの格納位置を特定し、主記憶に読み込むことができるように構成されている。

【0061】カーネルローダは、まず、カーネルにパラメータとして渡す主記憶リスト801、ロードモジュールリスト804、および、デバイスリスト802を初期化し、カーネル用のページテーブル領域を割り当てる(ステップ601)。

【0062】主記憶リスト801は、主記憶装置102の利用状況を示すデータ構造であり、カーネルローダが以降の処理で物理メモリの割り当てをする場合は、主記憶リスト801を参照、及び、変更して実施する。

【0063】次に、カーネル用のハードウェア構成の検査(ステップ602)、および、ハードウェア構成データの作成(ステップ603)を実施する。ステップ602においては、計算機100にどのようなハードウェアデバイスが接続されているかをハードウェアが検査する。続くステップ603では、ステップ602の結果に基づいてハードウェア構成に関するデータ構造であるデバイスリスト802が作成される。オペレーティングシステムのカーネルは、このデバイスリスト802を参照してカーネル初期化を実施する。

【0064】次に、オペレーティングシステムの構成情報700を磁気ディスク装置112より読み込み、パラメータテーブル810に構成情報のアドレスを設定する(ステップ604)。オペレーティングシステムのカーネルは、カーネル本体のファイルと、その他のデバイスドライバのファイルといったように、複数のファイルから構成されていてもよい。構成情報700は、予め決められたファイル名で磁気ディスク装置112に格納されており、ロードプログラムはこれを見つけることができる。

【0065】本実施形態におけるカーネル構成情報のデータ構造を図7に示す。

【0066】カーネル構成情報700は、カーネルローダやオペレーティングシステムが参照するデータを格納している。格納されているデータには名前が付けられており、プログラムは名前からそれに対応するデータを取得することができる。図7の例では、名前がメモリサイズというエントリ701があり、その内容としてメモリサイズを示すデータが格納されている。また、名前がsecondary OSというエントリ705は、第二のOS用のデータが格納されている。

【0067】カーネル構成情報700を読み込んだ後、カーネルローダは、カーネル構成情報700中のメモリサイズ701に格納されたデータの値によって、第一のOSに割り当てる物理メモリサイズを決定する(ステップ605)。次に、オブジェクトファイル703に設定されているカーネル構成ファイルのすべてを主記憶装置102に読み込み、ロードモジュールリスト804にエ

ントリを追加する(ステップ606~608)。ここではkernel、driver1、および、driver2というファイル名のオブジェクトファイルをロードする。

【0068】次にステップ605で求めたメモリサイズに対応するカーネル用のページテーブルを設定する(ステップ609)。ステップ609で設定したページテーブルのアドレスをページテーブルレジスタ105に設定し、プロセッサを仮想アドレスモードに移行させ(ステップ610)、構築した主記憶リスト801、デバイスリスト802、カーネル構成情報テーブル803、および、ロードオブジェクトリスト804の組からなるパラメータテーブル810をパラメータとして、カーネルの初期化ルーチンに制御を渡す(ステップ611)。カーネルのエントリポイントはカーネルファイル内のデータに記録されている。

【0069】次に、ステップ601から始まるブート手順が作成するハードウェア構成データと、ロードオブジェクトデータの構成について説明する。図8は、ハードウェア構成データとロードオブジェクトデータの構成を示す図である。

【0070】パラメータテーブル800は、カーネルローダが作成するデータ構造である。パラメータテーブル800から始まる3つのリストは、ローダが構築するカーネルの仮想空間に配置されており、カーネルから参照可能である。

【0071】パラメータテーブル800は、ローダが構築した3つのリストの先頭へのポインタと、1つのテーブルへのポインタを保持している。3つのリストとは、主記憶リスト801、デバイスリスト802、および、ロードオブジェクトリスト804で、1つのテーブルはカーネル構成情報テーブル803である。それぞれについて説明する。

【0072】主記憶リスト801は、主記憶ブロック記述データ810のリストである。主記憶ブロック記述データ810は、ベースアドレス811、ブロックサイズ812、ブロック利用状況813、および、次の主記憶ブロック記述データへのポインタ814から構成されている。

【0073】主記憶ブロック記述データ810は、ある連続した主記憶領域についての利用状況を記録している。ベースアドレス811は連続領域の開始物理アドレスを示し、ブロックサイズ812は連続領域の大きさを格納し、ブロック利用状況813は、当該連続領域が未使用であるか、あるいは、ローダにより割り当て済みであることを示す値が格納されている。そして、次エントリへのポインタ814によりリストを構成している。図8では、810の次のエントリは820である。主記憶リスト801を参照することで、物理メモリの利用状態を知ることができる。

【0074】デバイスリスト802は、カーネルローダが検出したハードウェアデバイスに関するデータを格納しており、ステップ603で作成される。デバイスリスト802は、デバイスデータ850からなるリストである。デバイスデータ850は、デバイスタイプ851、デバイス情報852、および、次のデバイスデータへのポインタ853から構成される。

【0075】デバイスタイプ851には、このデバイスデータエントリ850により記述されるデバイスの種類を示す値が格納されている。デバイス情報852は、デバイスの種類に特有なデータを格納している。例えば、割り込み番号やI/Oアドレスなどがそれに相当する。そして、次のエントリへのポインタ853によりリストを構成している。

【0076】カーネル構成情報テーブルへのポインタ803は、カーネルローダが主記憶装置102に読み込んだカーネル構成情報ファイル700の内容を指し示している。

【0077】ロードオブジェクトリスト804は、カーネルローダが主記憶にロードしたオブジェクトファイルに関するデータを保持している。ロードオブジェクトリストは、ロードオブジェクトデータ830のリストである。ロードオブジェクトデータ830は、オブジェクトファイル名831、オブジェクトアドレス832、および、次のロードオブジェクトデータへのポインタ833から構成される。

【0078】オブジェクトファイル名831は、ロードオブジェクト830により記述されているオブジェクトファイルのファイル名である。オブジェクトアドレス832は、当該オブジェクトファイルのヘッダ領域がロードされているカーネル空間のアドレスを格納している。そして、次のエントリへのポインタ833によりリストを構成している。

【0079】ロードオブジェクトリスト804は、カーネルローダがカーネルを構成するオブジェクトファイルを読み込むときに同時に作成される（ステップ608）。

【0080】次に、本実施形態におけるデバイスドライバのロード処理手順について説明する。デバイスドライバとは、計算機100に接続されたハードウェアデバイスを制御するためのソフトウェアであり、図6で説明したオペレーティングシステムのカーネルの初期化中、あるいは、オペレーティングシステムが立ち上がった後にロードできるものとする。

【0081】まず、デバイス管理テーブルについて説明する。図9は、第一のOSのデバイス管理テーブルの構造を示した図である。デバイス管理テーブルは、割り込み管理テーブル900と、I/Oアドレス管理リスト910の2つのデータ構造からなる。

【0082】割り込み管理テーブル900は、プロセス

サ101が受け付ける各割り込み番号について、第一のOSがその割り込み番号を利用するかどうかを示す値が格納されている。カーネルは、デバイスドライバがロード時に割り込み番号を要求した場合に、このテーブル900を検査し、使用されていない場合のみ、要求された割り込み番号を使用する権利をデバイスドライバに与える。既に利用済みであると記されている場合は、そのハードウェアデバイスは第一のOSからは利用できないことになる。例えば、割り込み番号3の場合、番号3に対応するデータ902が割り当て済みかあるいは未使用かを示す値を保持する。

【0083】I/Oアドレス管理リスト910についても同様である。I/Oアドレス管理リスト910は、I/Oアドレスを表現するエントリ920からなるリストである。エントリ920は、第一のOSが利用するI/Oアドレス921と、リストを構成するための次のエントリへのポインタ922からなる。割り込みベクタ管理テーブル900と同様、デバイスドライバがI/Oアドレス範囲を要求した場合、カーネルは、そのアドレス範囲が既に利用されているかI/Oアドレス管理リスト910により検査し、未使用である場合、このリスト910にエントリを追加して、利用許可を与える。

【0084】図10は、デバイスドライバのロード処理手順を示すフローチャートである。

【0085】デバイスドライバのロード処理では、まず、デバイスドライバの情報が取得される（ステップ1001）。デバイスドライバの情報とは、デバイスドライバのオブジェクトファイル名であり、デバイスドライバがオペレーティングシステムのカーネル初期化時にロードされる場合は、カーネル構成情報ファイル700のエントリ名オブジェクトファイル703から得られ、また、カーネル立ち上げ後にデバイスドライバがロードされる場合は、オペレータがロードを指示したオブジェクトファイル名が使用される。

【0086】次に、指定されたオブジェクトファイルを調べ、当該オブジェクトをロードするために必要なメモリサイズを調べる（ステップ1002）。調べたメモリサイズに必要な主記憶装置102上のメモリをオペレーティングシステムに対して要求し（ステップ1003）、当該デバイスドライバをメモリに読み込む（ステップ1004）。

【0087】次に、当該デバイスドライバのオブジェクトファイルからデバイスドライバの初期化処理のエントリポイントを得て、デバイスドライバの初期化処理を実行する（ステップ1005）。ステップ1005の初期化処理では、デバイスリスト802を参照することにより、当該デバイスドライバが対象とするハードウェアデバイスが計算機100に接続されているか確認する（ステップ1006）。

【0088】当該ハードウェアデバイスが接続されてい



るならば、当該ハードウェアデバイスの初期化、および、デバイスドライバの処理に必要なテーブルの初期化を実行する（ステップ1007）。もし、当該ハードウェアデバイスが割り込み番号あるいはI/Oアドレスを必要とするならば、当該デバイスドライバからオペレーティングシステムに対して、割り込み番号あるいはI/Oアドレス範囲を使用することを通知する。通知を受けたオペレーティングシステムは、図9の割り込み管理テーブル900とI/Oアドレス管理リスト910を調べ、要求された資源が利用可能な場合、デバイスドライバに使用の許可を与え、また、デバイスドライバ管理テーブルを更新することによって、他のデバイスドライバはこの資源を使用しないように設定する（ステップ1008）。割り込み番号の使用の許可が得られた場合、割り込みを処理するための割り込みハンドラ処理のアドレスをオペレーティングシステムに対して登録する（ステップ1009）。また、必要に応じて、当該デバイスドライバに固有の処理を実行する（ステップ1010）。

【0089】デバイスドライバのロード処理では、ステップ1001から1009の処理がすべて正しく行われた場合、ロードオブジェクトリスト804に当該デバイスドライバのオブジェクトファイル情報を追加し（ステップ1011）、当該デバイスドライバがオペレーティングシステムの一部となって動作し、ハードウェアデバイスを制御する。

【0090】図11は、デバイスドライバのアンロード処理のフローチャートである。

【0091】デバイスドライバのアンロードは、ハードウェアデバイスの使用を止めるために行うもので、オペレーティングシステム立ち上げ後、ロードされたデバイスドライバを、指示によりアンロードする場合や、ハードウェアデバイスが計算機から取り除かれた場合に実行される。

【0092】デバイスドライバのアンロード処理では、まず、オペレーティングシステムにより当該デバイスドライバの終了処理が呼び出される（ステップ1101）。

【0093】デバイスドライバの終了処理では、割り込みハンドラの登録、および、使用していた割り込み番号あるいはI/Oアドレス範囲の使用が解除される（ステップ1102、1103）。これにより、解除された割り込み番号あるいはI/Oアドレス範囲は未使用となり、他のデバイスドライバでの利用が可能になる。次に当該デバイスドライバが使用していたメモリを解放し（ステップ1104）、デバイスドライバの終了処理が完了する。

【0094】デバイスドライバの終了処理後、オペレーティングシステムは、デバイスドライバのオブジェクトファイルをロードしていた主記憶装置102上のメモリを開放し（ステップ1105）、ロードオブジェクトリ

スト804から当該デバイスドライバのオブジェクトファイルのリストを削除する（ステップ1106）。

【0095】次に、1台のPC上で複数のオペレーティングシステムを実行するための処理手順について説明する。本実施形態では、マルチOSを実行するための制御を第一のOSのデバイスドライバ（以下ではマルチOSドライバと記す）として組み込むものとする。

【0096】図12にマルチOSドライバの処理構造を示す。

【0097】マルチOSドライバの構造は大きく2つに分かれ、オペレーティングシステムから呼び出しによって実行される入り口関数部分1201と、入り口関数部分1201からの呼び出しによって実行される実行関数部分1202で構成される。

【0098】マルチOSドライバのすべての入り口関数は対応する実行関数のアドレスを保持しており、オペレーティングシステムから入り口関数が呼び出されると、対応する実行関数を呼び出し（ステップ1203）、呼び出された実行関数が当該関数の実際の処理を実行する（ステップ1204）。マルチOSドライバがこのように2つに分かれた構造になっている理由は、オペレーティングシステムがデバイスドライバをロード後、さらに、実行関数部分1202のアドレスを変更可能とするためである。

【0099】図13にマルチOSドライバのロード処理手順のフローチャートを示す。マルチOSドライバもデバイスドライバの一種であり、オペレーティングシステムは図10で説明したフローチャートに従ってマルチOSドライバのオブジェクトファイルをロードする。このため、ここでは、マルチOSドライバのデバイスドライバ固有処理（ステップ1010）の詳細のみを示している。

【0100】図13において、まず、すべてのオペレーティングシステムから参照可能な共通領域用の物理メモリを確保する（ステップ1301）。この共通領域には、マルチOSドライバの一部として、割り込みテーブル107、割り込み管理プログラム、割り込みハンドラ、各オペレーティングシステムから呼びだし可能なインターフェイスモジュール等が格納される。共通領域用の物理メモリの確保は、主記憶リスト801を参照し、空き領域を確保する。本実施形態では、物理メモリは第一のOSがロードされているだけであり、空いている物理メモリのうち、アドレスの大きいほうのメモリを必要な容量だけ確保する。

【0101】次に、共通領域のための、第一のOSの仮想アドレス空間を確保する（ステップ1302）。この仮想アドレス空間はすべてのオペレーティングシステムで一致させる必要があるため、予めすべてのオペレーティングシステムで取り決めたアドレスを割り当てるものとする。本実施形態では、第1のOSの仮想アドレス空

間の中で最もアドレスの大きい空間から必要な容量だけ確保する。

【0102】物理メモリと仮想アドレス空間の領域が確保できたならば、ステップ1302で確保した仮想アドレスからステップ1301で確保した物理メモリにアクセスが可能なように、第一のOSのページテーブルを設定する(ステップ1303)。

【0103】次に、マルチOSドライバが第一のOSがロードされた領域のオブジェクトとデータを、共通領域に複写し(ステップ1304)、図12で説明したマルチOSドライバ関数のうちのオペレーティングシステムから呼び出される入り口関数が保持する実行関数のアドレスを共通領域に複写した実行関数のアドレスに更新する(ステップ1305)。この実行関数のアドレスの変更により、第一のOSからマルチOSドライバの関数を呼び出した場合、共通領域にある実行関数のオブジェクトが実行されることになる。

【0104】図14に、仮想アドレス空間と主記憶装置102の物理メモリの対応を示す。

【0105】1401は物理メモリを、1402は第一のOSの仮想アドレス空間を示しており、図中、下部から上部に向かってアドレスが大きくなっていくものとする。第一のOSの仮想アドレス空間1402は、オペレーティングシステムのみがアクセス可能なカーネル空間1404とオペレーティングシステムとユーザプログラムがアクセス可能なユーザ空間に分かれる。カーネル空間1404の下位のアドレス側1408は、物理メモリ1401のうち、第一のOSに割り当てたメモリ1409にアクセスできるようにページマップテーブルが設定されていることを示している。

【0106】マルチOSドライバは、第一のOSによって、物理メモリ1409の一部の領域1413にロードされ、仮想アドレス空間1404のマルチOSドライバA領域1412にマップされる。マルチOSドライバA領域1412は、オペレーティングシステムの仮想アドレス空間の利用状況によって変化するため、予め予測することはできない。

【0107】マルチOSドライバは、すべてのオペレーティングシステムで使用するオブジェクトとデータを含むため、すべてのオペレーティングシステムで確保でき、かつ、一致した仮想アドレス空間の領域(マルチOSドライバB領域1415)にオブジェクトとデータを移動させる。これにより、すべてのオペレーティングシステムで共通のアドレスでマルチOSドライバにアクセスできることになる。また、マルチOSドライバB領域1415の物理メモリも、第一のOSに割り当てた物理メモリ内の領域1413から共通領域203に移動させることによって、オペレーティングシステムの不具合による誤ったメモリアccessを防ぐことができる。

【0108】図15は、第二のOSのロード手順を示す

フローチャートである。

【0109】この処理では、まず、第二のOS用に割り当てた物理メモリ領域に、第二のOSのオブジェクトファイルを読み込む必要がある。しかし、第二のOSの物理メモリ領域は、そのままでは第一のOSから書き込むことはできないので、割り当てた物理メモリ領域を第一のOSの仮想空間に一時的にマッピングする(ステップ1501)。

【0110】ステップ1502では、マッピングした領域に、第一のOSのファイル読み込み手順を利用して、第二のOSのオブジェクトファイルを読み込む。

【0111】次に、第二のOS用のページテーブルを作成する(ステップ1503)。このページテーブルも第二のOS用の領域に作成する。このとき共通領域203の領域についても、第二のOSの空間から参照できるように、ページテーブルを構築する。

【0112】共通領域203をマッピングするマルチOSドライバC領域1416は、第一のOSのマルチOSドライバB領域1415の仮想アドレスと同じ値になるように設定する。

【0113】次に、第二のOSからマルチOSドライバC領域1416に配置したマルチOSドライバのインタフェースモジュールの呼び出しができるように、インタフェースモジュールのアドレスを第二のOSの領域に書き込み、起動時の引数として第二のOSに渡す(ステップ1504)。同様に、マルチOSドライバから第二のOSを呼び出すときに必要な関数のアドレスを、オブジェクトファイルから取得し、共通領域203に保持しておく(ステップ1505)。

【0114】これで、第二のOS領域の設定を終了し、第一のOSにマッピングした、第二のOS用の物理メモリ領域のマッピングを解除する(ステップ1506)。

【0115】次に、OSコンテキストテーブル1610の第二のOSのコンテキストと、OS識別変数1630を設定する(ステップ1507)。OSコンテキストは、実行オペレーティングシステムを切り替えるときに参照するデータ構造で、ページテーブルアドレス値とスタックポインタの初期値とで構成される。ここでは、ページテーブルレジスタ値として第二のOSをマップするページテーブルのアドレスを、スタックポインタ値として第二のOSのカーネルスタックの初期アドレスを設定する。OS識別変数1630には、第一のOSが実行中であることを示す値が格納される。OSコンテキストテーブル1610とOS識別変数1630については後述する。

【0116】次に、第二のOSの初期化モジュールを実行する(ステップ1508)。これには、オペレーティングシステム空間の切り替えが伴う。オペレーティングシステムの切り替えについては、別のフローチャートにより説明する。

【0117】最後に、ステップ1509にて、現在の割り込みテーブルレジスタ104に登録されている第一のOSの割り込みハンドラのアドレスを、割り込み識別テーブル1620のハンドラ欄1622に複写し、割り込みテーブルレジスタ値を、マルチOSドライバに割り当てた割り込みテーブルのアドレスに変更する。これはプロセッサ101の割り込みテーブルレジスタ104の変更により実施する。

【0118】割り込みテーブルをマルチOSドライバ内のテーブルに変更するのは、割り込み発生時にどのオペレーティングシステムが実行していても、常にプロセッサ101の仮想アドレス空間に割り込みテーブルが存在している必要があるためである。割り込みテーブルに登録される割り込みハンドラも、マルチOSドライバ内に配置される。マルチOSドライバの領域は、ステップ1503にて、第二のOSの仮想空間にもマッピングして共通領域203とするので、いつでも参照できることになる。マルチOSドライバの割り込み処理については後述する。

【0119】図16は、共通領域203のうちデータ領域1600に格納されるデータのデータ構造を示した図である。

【0120】1610は、OSコンテキストテーブルである。OSコンテキストテーブル1610は、第一のOSと第二のOSとの間の切り替えに必要なデータを保持する。本実施形態では、第一のOSは第二のOSがアイドル状態の時のみ走行できるとする。この場合、第一のOS実行中のある時点で第二のOSへの切り替えが起こり、第二のOSの実行が終了した時点で、第一のOSに制御を戻せばよい。

【0121】したがって、それぞれで保存しておかなければコンテキストは一組でよい。第一のOSのコンテキストについては、OS切り替えが要求された時点でのページテーブルレジスタ値1611と、スタックポインタ値1612を保存しておけば、第二のOS実行終了後に、第一のOSに制御を復帰させることができる。

【0122】第一のOSから第二のOSへ制御を切り替えるときには第二のOSは動作していない。したがって、第二のOSのコンテキストは、ページテーブルアドレスもスタックポインタも固定のアドレスでよい。第二のOSのページテーブルレジスタ1613とスタックポインタ値1614は、第二のOSをロードするときに設定される(ステップ1507)。

【0123】1620は、割り込み識別テーブルである。割り込み識別テーブル1620には、外部割り込みの割り込み番号毎に、どのオペレーティングシステムが割り込み処理を処理するかを示す値1621と、割り込みハンドラのアドレス1622が記録されている。外部割り込みが発生すると、共通領域203内の割り込みハンドラが割り込みを捕獲し、この割り込み識別テーブル

1620の処理OS1621を参照して、どのオペレーティングシステムに処理させるかを決定し、ハンドラ1622のアドレスに制御を渡す。割り込み処理の詳細については、後述する。

【0124】1630は、実行中のオペレーティングシステムを示す値を格納しているOS識別変数である。OS識別変数1630は、ステップ1701から始まるOS切り替え手順でOS切り替えの度に設定される。割り込み処理では、OS識別変数1630を参照して割り込み処理手順が決定される。

【0125】1640は、第二のOSの実行中に、第一のOSが管理しているデバイスの割り込みが発生したかを示す遅延割り込み状態変数である。遅延割り込み状態変数1640は、どの割り込み番号の割り込みが発生したかを記録している。OS切り替え手順は、第二のOSの実行が終了したときに遅延割り込み状態変数1640を検査して、割り込み処理を起動するかを決定する(ステップ1708)。

【0126】1660は、第二のOSで動作するデバイスドライバが、割り込み番号あるいはI/Oアドレス範囲の資源を使用要求あるいは使用解除要求したことを示す変数である。第二のOSが実行中に資源が要求されたとき、第一のOSでこの資源を使用しないように通知する必要があるが、第二のOSから第一のOSの処理を直接は呼び出すことはできない。このため、マルチOSドライバが第二のOSからの資源の要求を受け、仮予約の状態を作り、第二のOSの実行が終了したときに、変数1660を検査することによって、マルチOSドライバが再度、第一のOSに対して資源の要求を行う。第二のOSが実行中は、第一のOSは必ず停止状態にあるため、資源がマルチOSドライバによって仮予約の状態でも、第一のOSがその資源を使用することはない。

【0127】図17は、本発明の実施の形態における、オペレーティングシステムの切り替え手順を示すフローチャートである。この切り替えの手順は、第一のOSの実行中に呼び出され、第二のOSへの切り替えを行う。

【0128】図17に示した手順は、第二のOSへの切り替え後に実行する第二のOSもモジュールのアドレスと、そのモジュールへ渡す引数を引数として受け取る。第二のOSのモジュールのアドレスは、ステップ1505で取得しているものとする。

【0129】まず、ステップ1701で、現在のスタックポインタ値とページテーブルレジスタ値を、OSコンテキストテーブル1610の、第一のOSのコンテキストとして保存する。ステップ1701では、現在のスタックポインタ値を1612に、現在のページテーブルレジスタ値を1611に保存する。他のレジスタコンテキストについては、OSコンテキストテーブル1610に保存する必要はない。必要があれば、第一のOSのスタックに保存すればよい。

【0130】スタックポインタ値とページテーブルレジスタ値を保存した後、ステップ1702にて、ページテーブルレジスタ105に第二のOSを仮想空間にマップするページテーブルのアドレスを設定する。これは、OSコンテキストテーブル1610の1613に記録されている。更に、スタックポインタを第二のOS用に設定する。これも、テーブル1610の第二のOSのスタックポインタ1614に格納されている。

【0131】次のステップ1703で、第一のOSの割り込み状態を示す遅延割り込み状態変数1640と割り込み資源の要求変数1660をクリアし、割り込み管理テーブル900とI/Oアドレス管理リスト910の内容を共通領域203に複写しておく。そして、現在実行中のOSを示しているOS識別変数1630を、第二のOSを示す値に書き換える(ステップ1704)。スタックポインタ、ページテーブルレジスタ105、および、OS識別変数1630は、常に一貫した値になっていなければならないので、ここまでのステップ1701、ないし、1704は、全て外部割り込みを禁止した状態で実行しなければならない。

【0132】続くステップ1705で、引数として渡されたモジュールのアドレスに制御を移し、第二のOSに制御を渡す。本実施形態においては、第一のOSは第二のOSが動作していないとき、つまり、第二のOSの処理が終了したときに、ステップ1706へ制御が戻る。

【0133】ステップ1706では、ステップ1701でOSコンテキストテーブル1610に保存したページテーブルレジスタ値1611と、スタックポインタ値1612のそれぞれを回復する。続くステップ1707で、OS識別変数1630を第一のOSが実行中であることを示す値に変更する。この2つステップの処理も割り込みを禁止した状態で実行しなければならない。

【0134】次に、第二のOSの実行中に発生した、第一のOSが管理するデバイスの外部割り込みを処理する。まず、ステップ1708では、遅延割り込み状態変数1640を検査して、割り込みが発生したかを検査する。発生していない場合は、ステップ1711を実行する。割り込みが発生している場合は、ステップ1709を実行する。このステップでは、第二のOSが実行中に発生した割り込みを、第一のOSが管理している遅延割り込み状態変数に、未処理の割り込みがある旨を記録する。続いて、第一のOSの割り込み処理を起動する(ステップ1710)。全ての割り込み処理が終了したとき、ステップ1711を実行する。

【0135】ステップ1711では、第二のOS実行中にデバイスドライバのロードあるいはアンロードによりハードウェアデバイスが必要とする割り込み番号あるいはI/Oアドレス範囲の資源の利用状態に変化があったかを検査する。つまり、第二のOSを実行中に、図10で説明した処理手順により、第二のOSが使用するハ

ドウェアデバイスを制御するため、対応するデバイスドライバのロードが行われ、資源の使用要求がデバイスドライバから第二のOSに通知された場合、第二のOSはマルチOSドライバに対して資源の使用が要求されたことを通知する。通知を受けたマルチOSドライバは、ステップ1703で共通領域203に複写した割り込み管理テーブル900とI/Oアドレス管理リスト910を調査し、資源の使用が可能ならば、共通領域203のテーブル(900, 910)を更新後、第二のOSに対して資源の使用の許可を出す。同時に、資源の使用の状態が変化したことを、変数1660に記録しておく。第二のOSを実行中にデバイスドライバのアンロードが行われた場合と同様であり、当該デバイスドライバから第二のOSへ、更に、第二のOSからマルチOSドライバへと要求が伝わり、変数1660に変化を記録する。

【0136】ステップ1711では、変数1660を検査することによって、割り込み番号あるいはI/Oアドレス範囲の資源の利用状態に変化があったことを検出し、変化があった場合は、共通領域203にある割り込み管理テーブル900とI/Oアドレス管理リスト910の内容を、第一のOSが管理する割り込み管理テーブル900とI/Oアドレス管理リスト910へと書き戻す(ステップ1712)。

【0137】これにより第二のOSが使用するハードウェアデバイスの資源は、第一のOSにも設定されるため、第一のOSのデバイスドライバにこの資源が割り当てられることが防げる。第一のOSから見て、第二のOSが使用するハードウェアデバイスの資源は、第一のOSにロードされているマルチOSドライバが使用する資源として見せることによって、オペレーティングシステム内の管理の一貫性は保たれる。

【0138】図18は、本実施形態の割り込み処理手順を示すフローチャートである。この手順を実行するモジュールは、割り込みハンドラとしてプロセッサ101の割り込みテーブル107に登録される。さらに、この割り込みハンドラは、どのオペレーティングシステムからも参照できる共通領域203に配置される。

【0139】外部割り込みが発生して、プロセッサ101により割り込みハンドラが起動されると、割り込みハンドラは割り込み要因を検査し、割り込みを発生したハードウェアデバイスが第一のOSが関するハードウェアデバイスか、第二のOSが管理するハードウェアデバイスかを判定する(ステップ1801)。この判定は、割り込み識別テーブル1620を割り込み番号をインデックスとしてOS欄1621を参照することにより実施される。第一のOSのハードウェアデバイスである場合はステップ1802へ、第二のOSのハードウェアデバイスである場合はステップ1805へ進む。例えば、図16でいえば、割り込み番号が1であれば第一のOSの割り込みであり、割り込み番号4であれば第二のOSの割り

込みとなる。

【0140】割り込みが第一のOSのハードウェアデバイスの割り込みである場合、ステップ1802を実行する。ステップ1802では、割り込み発生時に実行していたオペレーティングシステムを判定する。この判定は、OS識別変数1630を参照して実施する。実行中のOSが第一のOSの場合はステップ1803へ、第二のOSの場合はステップ1804へ進む。

【0141】ステップ1803から始まる処理は、第一のOSが管理しているハードウェアデバイスが、第一のOSを実行中に割り込みを発生した場合の処理である。

【0142】ステップ1803では、あたかも1801から始まる処理が存在せず、第一のOSの割り込みハンドラが、直接プロセッサ101から制御を受けたように見えるようにコンテキストを設定する。ここでコンテキストとは、スタックの内容やレジスタの内容を示す。そして、第一のOSの割り込みハンドラのアドレスは、割り込み識別テーブル1620のハンドラ欄1622に格納されている。例えば、割り込み番号1の割り込みならば、1をインデックスとして割り込み識別テーブルを参照して、ハンドラのアドレスを求める。この場合、ステップ1801から始まる手順には制御は戻らず、第一のOSが処理を続ける。

【0143】第一のOSが管理しているハードウェアデバイスが、第二のOSを実行中に割り込みを発生した場合、ステップ1804を実行する。ステップ1804では、遅延割り込み状態変数1640に割り込みを発生したハードウェアデバイスの割り込み番号を記録する。割り込みハンドラの処理はこれで終了する。この場合の割り込みの処理は、実行OSが第一のOSに切り替わったときに実行される(ステップ1708)。

【0144】発生した外部割り込みが、第二のOSが管理するハードウェアデバイスの割り込みだった場合、ステップ1805へ進む、どちらのOSが実行中であるかを検査する。ここでも、OS識別変数1630によって実行中のOSを判定する。第一のOSが実行中の場合は、ステップ1806へ、第二のOSが実行中の場合はステップ1811へ進む。

【0145】第二のOSが管理するハードウェアデバイスの割り込みが、第二のOSの実行中に発生した場合、ステップ1811を実行する。ステップ1811は、第二のOSの割り込みハンドラを起動する。第二のOSの割り込みハンドラのアドレスは、割り込み識別テーブル1620のハンドラ欄1622に記録されている。第二のOSの割り込みハンドラ処理が終了して戻ってきたら、この割り込みハンドラも終了し、割り込まれたときのコンテキストを回復して制御を戻す。

【0146】第二のOSが管理するハードウェアデバイスの外部割り込みが、第一のOSの実行中に発生した場合、ステップ1806を実行する。この場合は、第一の

OSの実行よりも第二のOSの処理を優先して実行する。

【0147】まず、ステップ1806では、コンテキストを保存する。ここでのコンテキストとは、割り込み処理が終了した後で第一のOSに制御を戻すときに、割り込まれたときの状態を回復するのに必要なスタックの内容とレジスタの内容を示す。このコンテキストは、第一のOSのカーネルのスタックに保存される。

【0148】続いて、実行OSの切り替えと第二のOSの割り込み処理の起動を実行する(ステップ1807、1808)。これは、ステップ1701から始まる手順により実行する。

【0149】第二のOSの処理が終了した時点で、第一のOSへの切り替えを実行し(ステップ1809)、割り込み時のコンテキストを回復し(ステップ1810)、第一のOSの処理を再開する。ステップ1809の処理は、必ずしも、ステップ1801から始まる処理と同一のもジュールで実行されなくてもよい。第一のOSへの切り替えにより処理はこのモジュールに復帰する。

【0150】2つのオペレーティングシステムで共有しているクロック割り込みの処理について説明する。

【0151】クロック割り込みは、共通領域203内の割り込みハンドラにより捕獲する。この割り込みのハンドラでは、まず、第二のOSのクロック割り込み用の割り込みハンドラを実行する。第二のOSの割り込みハンドラはハンドラ欄1623に格納されている。第二のOSの割り込みハンドラが終了したら、図18のステップ1802から始まる処理により第一のOSの割り込み処理を実行する。第1のOSの割り込みハンドラのアドレスはハンドラ欄1622に格納されている。

【0152】以上により、一台の計算機で2つのオペレーティングシステムを同時に動作させることが可能になる。

【0153】さらに、上述した実施形態において、図15で示した第二のOSのロード手順に図19のフローチャートで示すようなステップ1510を加えることによって、第二のOSに第一のOSの動作状態を監視させることが可能になる。図19の処理手順について以下説明する。

【0154】図19の処理手順のうち、ステップ1501からステップ1509までは、図15のフローチャートの処理と同じであり、新たに監視処理を実施するためのステップ1510を追加している。

【0155】オペレーティングシステムは、実行中に障害が発生し、処理の継続が不可能と判定した場合、オペレーティングシステムのエラー処理関数を実行することによって、計算機に接続している全てのハードウェアデバイスを停止させ、最後に計算機のプロセッサを停止させる。本実施形態では、一台の計算機上で複数のオペレーティングシステムを動作させているため、一つのオペ

レーティングシステムの障害で、必ずしも計算機全体を停止させる必要はなく、障害を起こしたオペレーティングシステム以外のオペレーティングシステムが計算機の動作を継続することが可能である。

【0156】本実施形態では、第一のOSに障害が発生し、第二のOSがこれを検出するものとする。ステップ1510では、第二のOSのローディング処理の最後に、第一のOSが呼び出すエラー処理関数の先頭の命令を、割り込みを発生する命令に書き換える。例えば、図16に示した割り込み識別テーブル1620の状態で、割り込み番号5を発生させる命令を書き込むと、第一のOSで継続不能な障害が発生した場合、第一のOSのエラー処理関数が実行される前に割り込み番号5がプロセッサ101で発生する。割り込み番号5は、第二のOSが管理する割り込みであり、割り込み番号5のハンドラ欄1623の内容にしたがって、第二のOSのハンドラが呼び出される。つまり、第一のOSのエラー処理関数が実行される前に、第二のOSは第一のOSに障害が発生したことを検知できる。割り込み番号5の割り込みハンドラ処理で、マルチOSドライバに対して、第一のOSがダウンしたことを通知し、以後、第二のOSのみを動作させるようにすれば、第二のOSの処理は継続させることができる。処理を継続する第二のOSが、第一のOSの障害診断や障害回復を行い、オペレータに対する通知を行うことが可能である。これにより、一台の計算機で複数のオペレーティングシステムを同時に動作させる環境において、1つのオペレーティングシステムが他のオペレーティングシステムで発生した障害を監視するシステムを実現できる。

【0157】また、上述した実施形態では、一台の計算機で複数のオペレーティングシステムを動作させる機能を、マルチOSドライバというデバイスドライバで実施したが、計算機のブート処理によりロードされるオペレーティングシステムの内部にマルチOSを実施する機能を組み込むことも可能である。この実施形態では、図6のブート処理のステップ611の処理の中で、図13に示したマルチOSを実施する処理(ステップ1301から1306)を実行することによって、第一のOSが立ち上がった時点で、第二のOSのローディングが可能な状態になる。この実施形態においても、図15、図16、図17、図18、図19に示した処理手順は変更することはない。これにより、1台の計算機で、自オペレーティングシステム以外のオペレーティングシステムを同時に動作させることができるオペレーティングシステムを実現し、複数のオペレーティングシステムを同時に動作させることができる。

【0158】

【発明の効果】以上説明した実施形態によれば、特別なハードウェアの支援なしで、1台の計算機で複数のオペレーティングシステムを同時に動作できる。第一のOS

に欠けている機能を容易に導入することができ、さらに、その機能を第一のOSとはまったく独立して動作させることができる。

【0159】また、マルチOSを実施する機能をデバイスドライバとして実現することにより、第一のOSの処理を変更することがないため、既存のオペレーティングシステムに対して容易に機能を追加できる。さらに、1台の計算機で複数のオペレーティングシステムを同時に動作させた環境で、1つのオペレーティングシステムは他のオペレーティングシステムの障害を監視することができ、監視オペレーティングシステムが被監視オペレーティングシステムの障害診断や障害回復を行うことによって、計算機全体の信頼性および保守性を向上させることができる。

【0160】さらにまた、ハードウェアデバイスが使用する割り込み資源を、複数のオペレーティングシステム間で動的に割り当てることによって、1つのオペレーティングシステムに割り当てたハードウェアデバイスを、該オペレーティングシステムが利用終了後、他のオペレーティングシステムが利用することが可能になる。

【0161】

【発明の効果】本発明によれば、特別なハードウェアを用いる必要なく、また、入出力処理等の特権命令のエミュレーションを行うことなく1台の計算機上で複数のオペレーティングシステムを同時、並列的に動作させることができる。

【図面の簡単な説明】

【図1】本発明の一実施形態における計算機構成を示すブロック図である。

【図2】計算機構成を示す模式図である。

【図3】ページテーブルの構成を示す図である。

【図4】割り込みテーブルの構成を示す図である。

【図5】割り込み制御装置の構成を示す図である。

【図6】計算機のブート処理手順を示すフローチャートである。

【図7】第一のOSのカーネル構成情報ファイルの構成を示す図である。

【図8】カーネル起動パラメータのデータ構造を示す図である。

【図9】第一のOSのデバイス管理テーブルのデータ構造を示す図である。

【図10】オペレーティングシステムのデバイスドライバのロード処理手順を示すフローチャートである。

【図11】オペレーティングシステムのデバイスドライバのアンロード処理手順を示すフローチャートである。

【図12】マルチOSドライバの処理構造を示す図である。

【図13】マルチOSドライバのロード処理手順のフローチャートである。

【図14】仮想アドレス空間と主記憶装置102の物理

メモリの対応を示す図である。

【図15】第二のOSのロード処理手順を示すフローチャートである。

【図16】第一のOSと第二のOSが共有するデータ構造を示す図である。

【図17】実行OSの切り替え処理手順を示すフローチャートである。

【図18】割り込み処理手順を示すフローチャートである。

【図19】第2のOSに監視機能を付加するための、第二のOSのロード処理手順を示すフローチャートである。

【符号の説明】

100…計算機

101…プロセッサ

102…主記憶装置

\* 103…演算装置

104…割り込みテーブルレジスタ

105…ページテーブルレジスタ

106…アドレス変換装置

107…割り込みテーブル

108…ページテーブル

109…バス

110…信号線

111…クロック割り込み生成器

112…割り込みコントローラ

113…キーボード

114…ディスプレイ

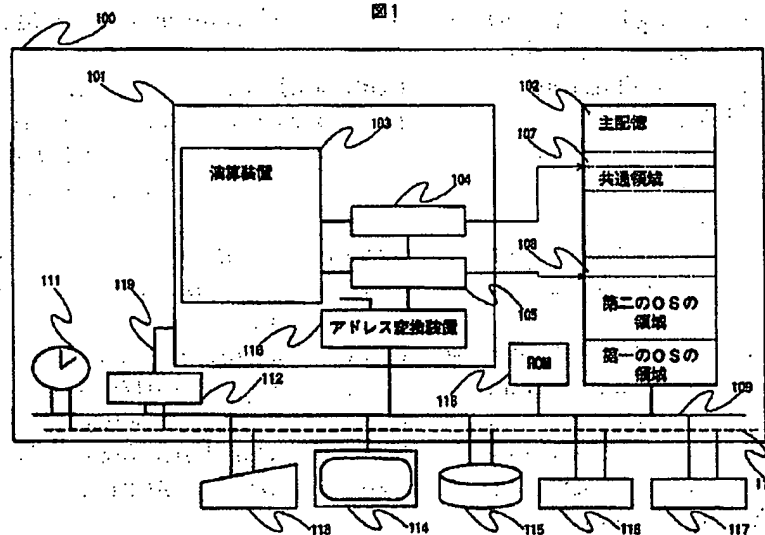
115…磁気ディスク

116、117…外部機器

118…記憶装置

\* 119…割り込みバス

【図1】



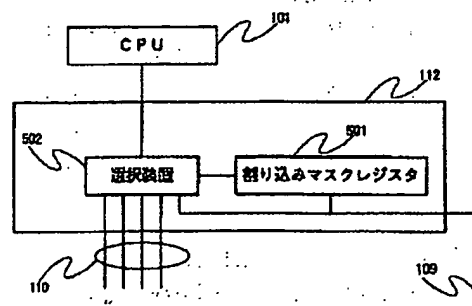
【図4】

図4

割込 番号	割り込みハンドラ開始アドレス
0	800h
1	830h
2	500h
3	
4	
5	

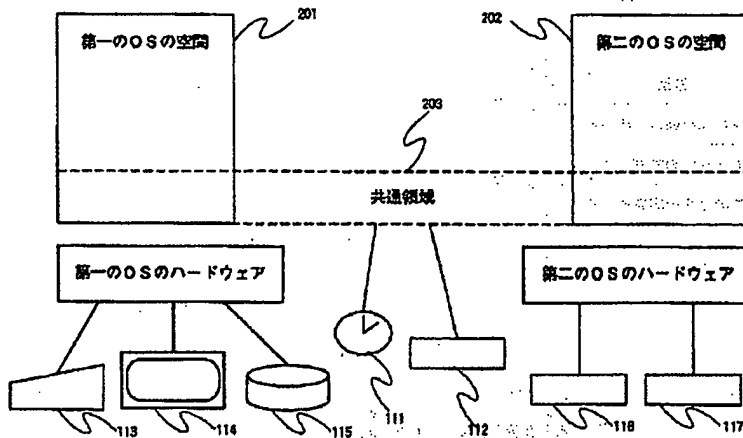
【図5】

図5



【図2】

図2



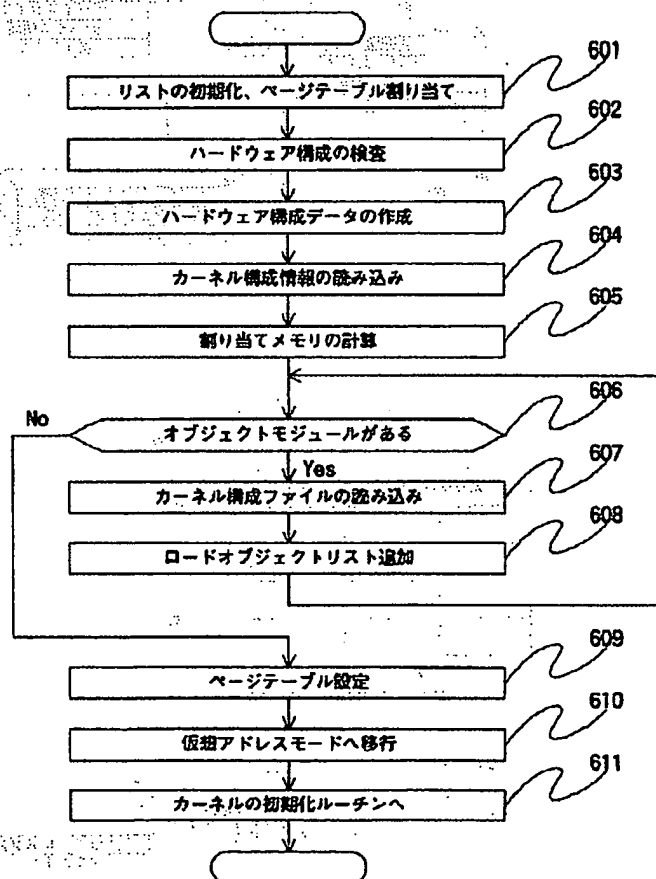
【図3】

図3

仮想ページ番号	有効	物理ページ番号
0	✓	0
1	✓	5 6
2	✓	2 3
3		
4		
5	✓	3 2
		⋮

【図6】

図6



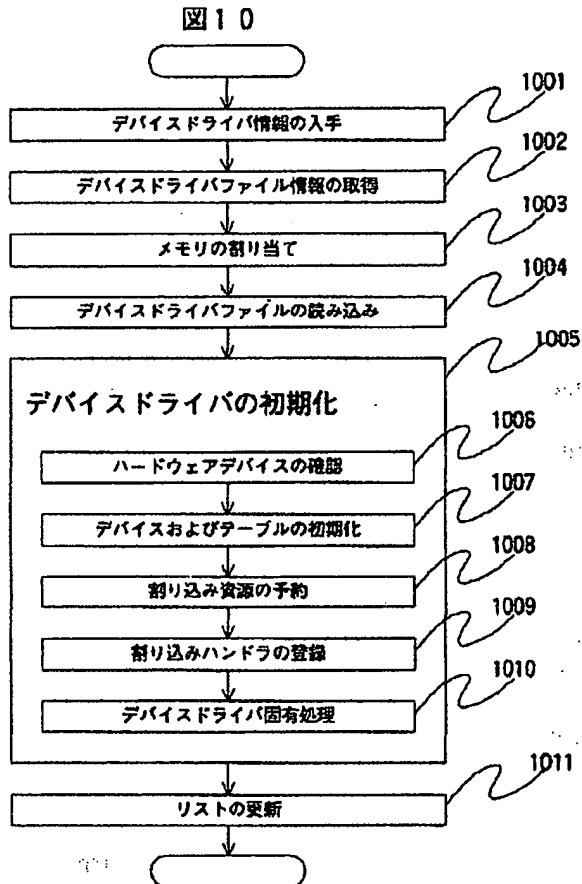


527

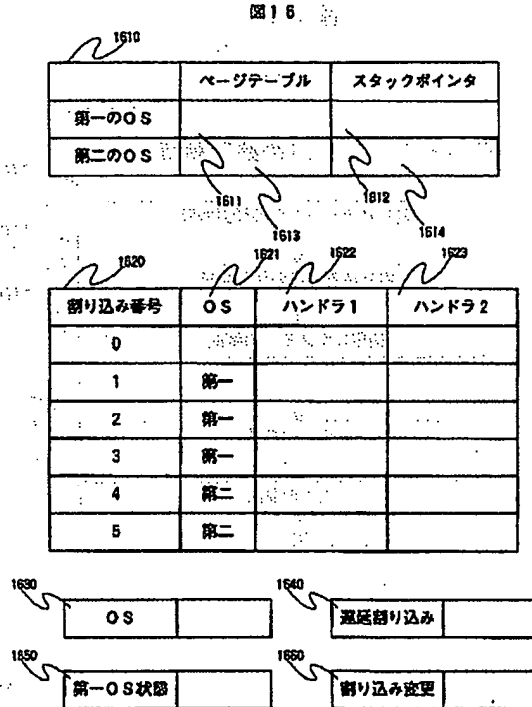
【图8】



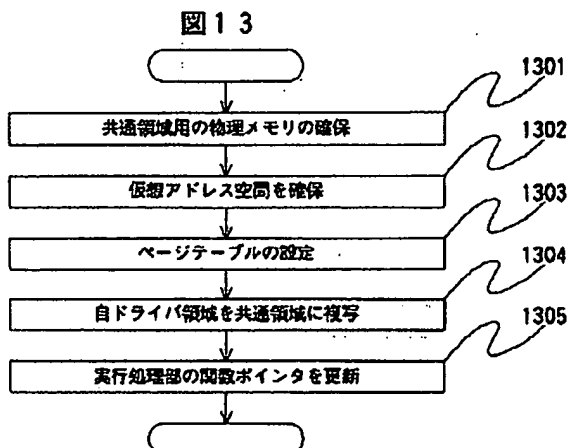
【図10】



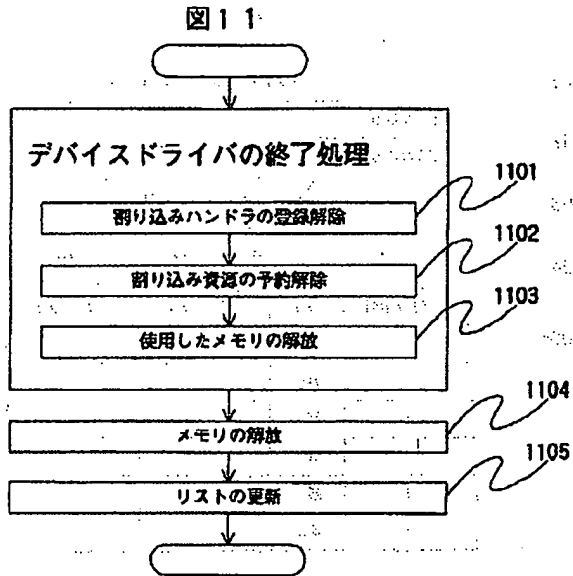
【図16】



【図13】

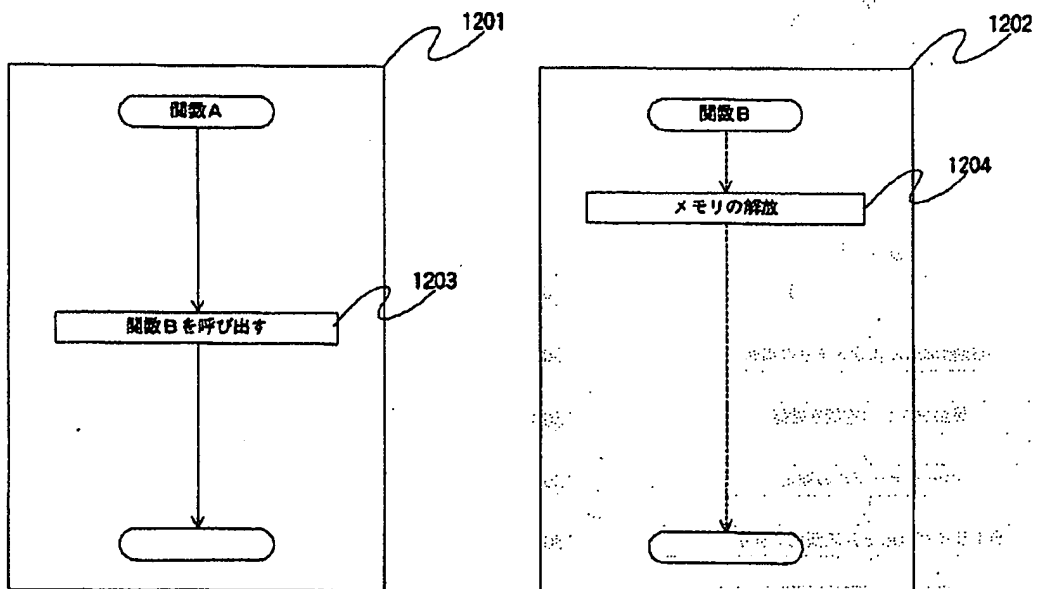


【図11】

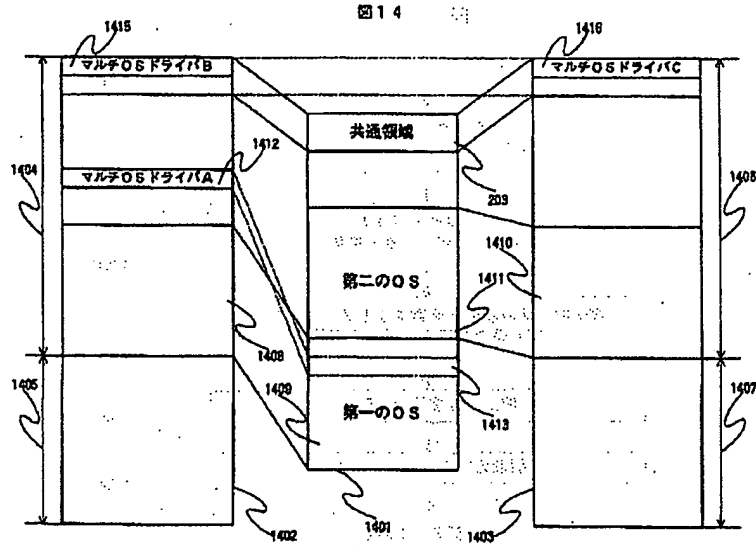


【図12】

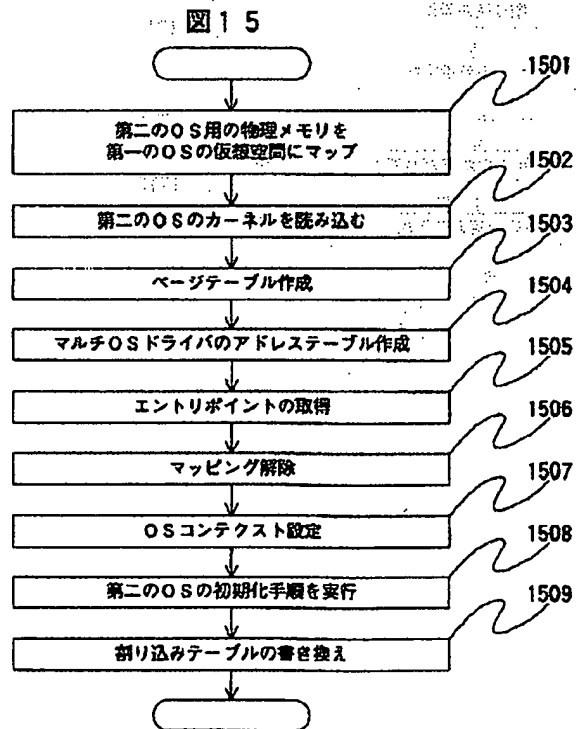
図12



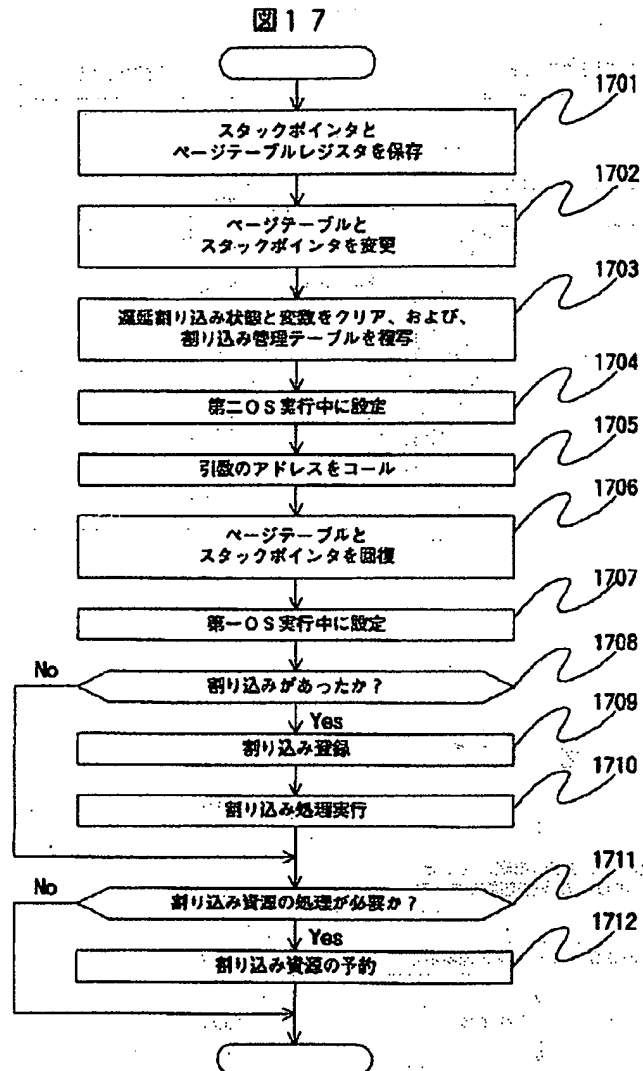
【図14】



【図15】

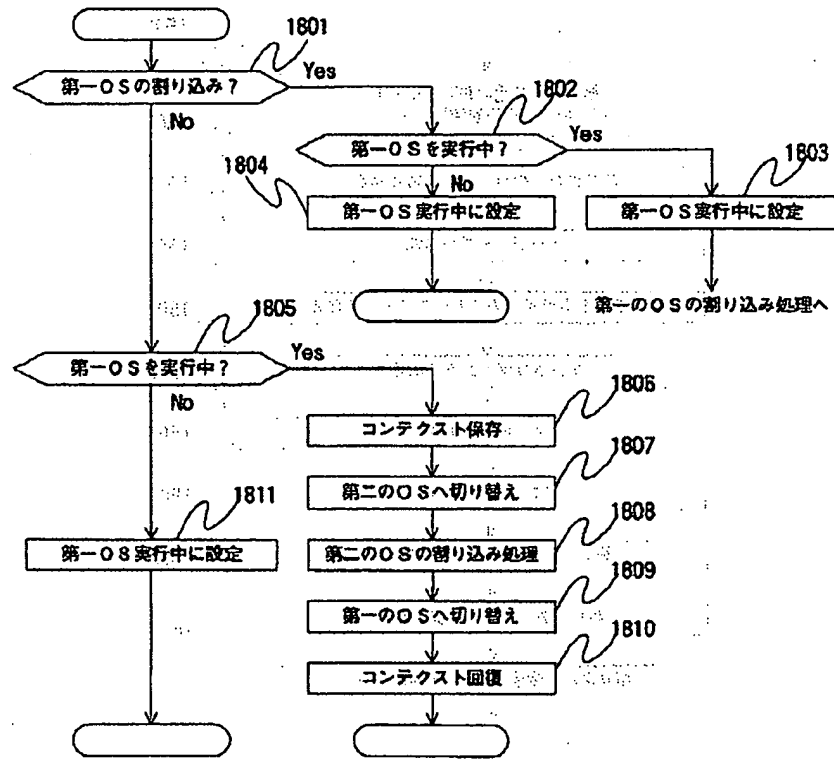


【図17】

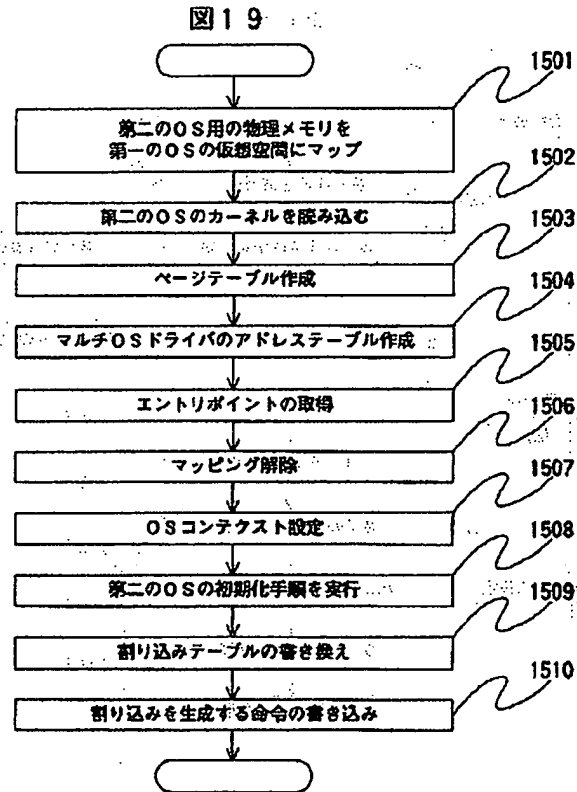


【図18】

図18



【図19】



フロントページの続き

(72) 発明者 佐藤 雅英  
 神奈川県川崎市麻生区王禅寺1099番地 株  
 式会社日立製作所システム開発研究所内  
 (72) 発明者 梅都 利和  
 愛知県尾張旭市晴丘町池上1番地 株式会  
 社日立製作所情報機器事業部内

Fターム(参考) 5B042 GA22 GA23 GA25 GC07 GC12  
 JJ01 JJ05  
 5B076 AA13 AA14 AB14 CA08  
 5B098 BA06 BB11 EE02 EE06 GA02  
 GC01 GD02 GD20 GD22 HH04  
 HH07 JJ08